YOUR FINAL TASK

COMPUTER PROGRAMMING 2
(Week 9)

NAME: **Aidre Love S. Cabrera** **GRADE & SECTION: 12 – STEM A**
TEACHER: **Antony Pleños** **DATE SUBMITTED: April 2, 2022**

A. Instructions: Write **True** on the blank provided if the statement is correct, otherwise, write **False**.

False 1. Classes that help handle errors in Java are called error classes.

True 2. It is possible to have several `catch` blocks following a `try` block.

False 3. A `try` block does not need to have a matching `catch` block.

False 4. A `catch` block does not need to have a matching `try` block.

True 5. Several `catch` statements following a single `try` statements should handle different exceptions.

False 6. The `finally` statement is required after using `try` and `catch` statements.

True 7. The block within the `finally` statement will be executed regardless of whether or not an error is encountered.

True 8. The `IOException` class handles errors that occur during input and output.

True 9. The `Exception` class handle all types of exceptions.

True 10. Declaring `catch` blocks before a block that handles an `Exception` class to handle other types of errors would be redundant.

B. Instructions: Insert the missing keyword to execute code, after try catch, regardless of the result.

Problem

```
try {
    int[] myNumbers = {1, 2, 3};
    System.out.println( myNumbers [10]);
} catch (Exception ) {
    System.out.println("Something went wrong.");
} finally {
    System.out.println("The 'try catch' is finished.");
}
```

Solution

Solution.java

```java
try {
  int[] myNumbers = {1, 2, 3};
  System.out.println(myNumbers[10]);
} catch (Exception e) {
  System.out.println("Something went wrong.");
} finally {
  System.out.println("The 'try 'catch' is finished");
}
```

C. Instructions: Give the output produced by the following code snippets.

```
1. int n = 5;
   try {
         n = n / 0;
   }
   catch (ArithmeticException e) {
         System.out.println("Arithmetic Exception Caught");
   }
   catch (NumberFormatException e) {
         System.out.println("Number Format Exception Caught");
   }
   finally {
         System.out.println("Done")
   }
```

Answer:

```
java: ‘;’ expected; Line #
```

Realistically, the provided code will result a syntax error. That is because in Line 12, in finally block, the print method has a missing semicolon. Therefore, the output that will be produced when strictly following the provided code snippet will be “java: “expected; Line 11.”

Proof of Error

Terminal Output

```java
// C1 Code Snippet
static void sectionC1() {
    int n = 5;
    try {
        n = n / 0;
    } catch (ArithmeticException e) {
        System.out.println("Arithmetic Exception Caught");
    } catch (NumberFormatException e) {
        System.out.println("Number Format Exception Caught");
    } finally {
        System.out.println("Done");
    }
}

/*

Terminal Output:
Arithmetic Exception Caught
Done

*/
```

To alleviate the simple error, we can simply add a semicolon before the error line and the output will be:

```
Arithmetic Exception Caught
Done
```

C2 Final Answer and Proof

Terminal Output

```java
// C1 Code Snippet
static void sectionC1() {
    int n = 5;
    try {
        n = n / 0;
    } catch (ArithmeticException e) {
        System.out.println("Arithmetic Exception Caught");
    } catch (NumberFormatException e) {
        System.out.println("Number Format Exception Caught");
    } finally {
        System.out.println("Done")
    }
}

/*

Terminal Output:
G:\Github Projects\src\fileContainers\week9.java:25:43
java: ';' expected;Line 11

*/
```

2.
```
int n = 5;
try {
      n = n / 0;
}
catch (Exception e) {
      System.out.println(“Exception Caught”);
}
finally {
      System.out.println(“Done”)
}
```

Answer:

```
java: ‘;’ expected; Line #
```

The same error is encountered in this snippet from the previous snippet. However, the solution is the same.

Proof of Error

C2 Final Answer and Proof

```
Exception Caught
Done
```

Appendix I

```java
package fileContainers;

public class week9 {
    static void sectionB() {
        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]);
        } catch (Exception e) {
            System.out.println("Something went wrong.");
        } finally {
            System.out.println("The 'try 'catch' is finished");
        }
    }

    static void sectionC1() {
        int n = 5;
        try {
            n = n / 0;
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception Caught");
        } catch (NumberFormatException e) {
                System.out.println("Number Format Exception Caught");
        } finally {
                System.out.println("Done");
        }
        }

    static void sectionC2() {
        int n = 5;
        try {
            n = n / 0;
        }
        catch (Exception e) {
            System.out.println("Exception Caught");
        }
        finally {
            System.out.println("Done");
        }
    }

    public static void main(String[] args) {
        System.out.println("Section B\n");
        sectionB();
        System.out.println("\nSection C, #1\n");
        sectionC1();
        System.out.println("\nSection C, #2\n");
        sectionC2();
    }
}
```

Appendix II

```
//Console Output from Intellij

/*

Section B

Something went wrong.
The 'try 'catch' is finished

Section C, #1

Arithmetic Exception Caught
Done

Section C, #2

Exception Caught
Done

Process finished with exit code 0

*/
```